Adachi uses a software emulator to emulate a second instruction set on a computer having a different native instruction set. An emulator is coded in the native instruction set and runs on the base hardware. When the emulator is running, the computer treats the emulated program, in a non-native instruction set, as it would treat any other data.

Adachi states in the abstract and in the Summary that he has processors to execute two different "instruction word sets." However, neither the figures nor the Description of the Preferred Embodiment discuss hardware for executing the "target" instruction set. In the Description, Adachi only describes that the "target" instruction set is emulated in software – see, *e.g.*, Fig. 1 and col. 3, lines 20-25, where the emulator is shown as a peer to "user programs" on the single processor. Adachi states that the essential problem he solves is the mismatch between the hardware and the target instruction set (col. 1, lines 15-19). This problem could not arise if there were hardware that directly executed the target program, without a software emulator. The most likely reading of Adachi '975 is that the Summary and Abstract were intended to explain that Adachi provides a hardware "processor" to execute one instruction set, and a software "processor" to execute the other.

Even if Adachi does teach two entirely different hardware processors for the two instruction sets, this would have no bearing on claims 1, 2 and 10 of this application. Each claim recites a single processor that combines several properties[1], not a mix-and-match combination of disparate processors, each of which has a few of the properties but no one of which has all.

The "table entry" referred to in col. 4, line 51, is a software structure established by the emulator for use by the emulator. See Adachi '975, col. 4, lines 46-51. The table is apparently not used to control Adachi's hardware.

The attached excerpt from an IBM Principles of Operation manual gives the definition of the IBM SVC instruction. (Even though this is the 1999 version of the book, Applicant observes that the definition of the SVC instruction has been essentially unchanged since the early 1960's.) The SVC instruction is defined to execute as follows (with a few intermediate steps left out):

(a) store the current PSW (a total of 64 bits, including program counter, condition codes, etc.) at real location 32 (this store bypasses address translation)
(b) Fetch a new PSW from real location 96 (this load bypasses address translation)

[1] The claims would, of course, literally read on a computer with a plurality of processors, where at least one of the processors possessed all of the properties recited in the claim.

(c) The 8 low-order bits of the SVC instruction are copied to the system interruption code and to real locations 138-139.

Note that no "table" appears in the description of the SVC instruction.

Rejection under § 103(a) of claim 2

Claim 2 was amended in the October Response. Before amendment, the claim had recited that the "table [is] addressed by the address of instructions executed;" the claim was amended independent of any statutory requirement to redundantly state in the final paragraph that the table entry on which action is based is the "table entry associated with the address of the instruction." As amended, claim 2 recites as follows:

> 2. A method, comprising the steps of:
>
> as part of the basic instruction cycle of executing an instruction of a non-supervisor mode program executing on a computer, consulting a table, the table being addressed by the address of instructions executed, for attributes of the instructions;
>
> controlling an architecturally-visible data manipulation behavior or control transfer behavior of the instruction based on the contents of a table entry associated with the address of the instruction.

Even before amendment, claim 2 recited that "the table [is] addressed by the address of instructions executed." This limitation is nowhere mentioned in the § 103 portion of the Office Action. Because the rejection is incomplete, Applicant respectfully observes that no *prima facie* case of obviousness has been raised.

Adachi's SVC instruction has an immediate field (Adachi '975, col. 4, lines 55-56) embedded in the instruction. *See also* IBM Principles of Operation. Adachi's software emulator uses that immediate as the index into a jump table (col. 4, lines 52-55). Adachi uses the <u>contents</u> of an instruction to index into the table, not "the <u>address</u> of instructions executed" (as recited in the claim). Because the claim recites an element entirely absent from the prior art, no *prima facie* obviousness rejection can be raised. MPEP § 2143.03.

Rejection under § 103(a) of claim 10

Claim 10 recites as follows:

> 10. A microprocessor chip, comprising:
>
> instruction pipeline circuitry;

table lookup circuitry designed to index into a table by a memory address of a memory reference arising during execution of an instruction, and to retrieve a table entry corresponding to the address, the table entry being distinct from the memory referenced by the memory reference;

the instruction pipeline circuitry being responsive to the contents of the table to alter a manipulation of data or transfer of control behavior of the instruction in a manner incompatible with the architectural definition of the instruction.

Claim 10 recites that hardware execution of the "instruction pipeline circuitry" is altered in "a manner incompatible with its architectural definition." This limitation is nowhere mentioned in the § 103 portion of the Office Action. Because the rejection is incomplete, Applicant respectfully observes that no *prima facie* case of obviousness has been raised.

Further, Adachi nowhere suggests that he alters the underlying hardware "in a manner incompatible with the architectural definition" of the computer. Adachi describes that his emulator (a software program) "detects" certain conditions (col. 4, line 46-50) about instructions of the target instruction set as those target instructions are emulated in software, and modifies the software emulation of those instructions by executing different software (col. 5, lines 4-10). But nothing in Adachi's disclosure suggests that any feature of hardware execution of any instruction executing in that hardware departs from the ordinary architectural definition.

Applicant further observes that the jump table referenced in Adachi '975 is used only by Adachi's software emulator, not by the hardware.

In contrast, claim 10 recites an instruction whose hardware execution of the "instruction pipeline circuitry" is altered in "a manner incompatible with its architectural definition." Because the claim recites an element entirely absent from the prior art, no *prima facie* obviousness rejection can be raised. MPEP § 2143.03.

Rejection under § 103(a) of claim 1

Even before amendment, claim 1 recites both that "the table having an entry associated with each corresponding address range," and "trigger[ing] an interrupt [for an instruction, the] architectural definition of the instruction not calling for an interrupt." Neither limitation is mentioned anywhere in the § 103 portion of the Office Action. Because the rejection is incomplete, Applicant respectfully observes that no *prima facie* case of obviousness has been raised.

Further, Applicant notes that the amendments to claim 1 were made merely as an accommodation to the Examiner, and are not to be construed as a concession that any rejection was properly raised under § 112 ¶ 2.

Conclusion

In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. In the event that an extension of time is required, Applicant petitions for that extension of time required to make this response timely. Kindly charge any additional fee, or credit any surplus, to Deposit Account 50-0324, Order No. 30585/16.

Respectfully submitted,

SHEARMAN & STERLING

Dated: January 9, 2001

By: Dvd E. Boundy
David E. Boundy
Registration No. 36,461

Mailing Address:
SHEARMAN & STERLING
599 Lexington Avenue
New York, New York 10022
(212) 848-4000
(212) 848-7179 Telecopier

7.5.94 SUPERVISOR CALL

```
SVC     I            [RR]


 ________ ________
| '0A'   |   I    |
|________|________|
0         8       15
```

```
The instruction causes a supervisor-call interruption, with the I field of
the instruction providing the rightmost byte of the interruption code.
```

Bits 8-15 of the instruction, with eight zeros appended on the left, are placed in the supervisor-call interruption code that is stored in the course of the interruption. See "Supervisor-Call Interruption" in topic 6.7.

A serialization and checkpoint-synchronization function is performed.

Condition Code: The code remains unchanged and is saved as part of the old PSW. A new condition code is loaded as part of the supervisor-call interruption.

Program Exceptions: None.

6.7 Supervisor-Call Interruption

The supervisor-call interruption occurs when the instruction SUPERVISOR CALL is executed. The CPU cannot be disabled for the interruption, and the interruption occurs immediately upon the execution of the instruction.

The supervisor-call interruption causes the old PSW to be stored at real location 32 and a new PSW to be fetched from real location 96.

The contents of bit positions 8-15 of the SUPERVISOR CALL instruction are placed in the rightmost byte of the interruption code. The leftmost byte of the interruption code is set to zero. The instruction-length code is 1, unless the instruction was executed by means of EXECUTE, in which case the code is 2.

The interruption code is placed at real locations 138-139; the instruction-length code is placed in bit positions 5 and 6 of the byte at real location 137, with the other bits set to zeros; and zeros are stored at real location 136.